REMARKS

This application has been carefully considered in connection with the Examiner's Office Action dated February 22, 2007. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 1-39 were pending at the time of the Office Action.

Claims 1-39 were rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1-3, 5-8, 10-37 and 39 were rejected under 35 USC § 103 as being unpatentable over "PERCobol Evaluation and review Guide: Getting Started" (hereinafter Getting_Started) in view of "PERCobol enterprise Edition V2.4: Programmer's Guide" (hereinafter Prog_Guide).

Claims 4 and 28 were rejected under 35 USC § 103 as being unpatentable over Getting_Stared and Prog_Guide, as applied to claims 1 and 25 and further in view of "Pro*COBOL Precompiler Programmer's Guide," March 2002, Oracle Corp., Part No. A9610901, pp. i-xi, "2-1" thorugh "2-32", and "12-1" through "12-22", (75 pages) (hereinafter Pro*C).

Claim 9 was rejected under 35 USC § 103 as being unpatentable over Getting_Started and Prog_Guide, as applied to claim 1, and further in view of U.S. Patent No. 5,146,593 to Brandle, et al. (hereinafter Brandle).

Summary of Response

Paragraph 001 has been amended to reflect the current U.S. Patent Application Serial Numbers.

Claims 1, 3-12, 14, 18, 23, and 35 were amended.

Claims 13, 15-17, 19-22, 24-34, and 37-39 remain as originally filed.

Claims 2 and 36 have been canceled.

Summary of Claims Pending:

Claims 1-39 are currently pending following this response.

Response to Rejections under Section 101

In the Office Action dated February 22, 2007, Claims 1-39 were rejected under 35 USC § 101 because the claimed invention is directed to non-statutory subject matter.

Claims 23-34 were rejected as elements that can be reasonably interpreted as software, wherein the claims do not define any structural and functional interrelationships between the software elements and a computer. Claim 23 has been amended here to further clarify the structural and functional interrelationships between the claimed compiler and a computer. In particular, claim 23 has been amended to recite that the compiler is stored on a computer-readable medium and therefore is a computer element which defines structural and functional interrelationships between the compiler and the rest of the computer which permit the compiler's functionality to be realized, and is thus statutory. See *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d

1031, 1035. Dependent claims 24-34 are therefore also directed to statutory subject matter.

Claims 1-39 were rejected as directed to processes wherein the claimed subject matter does not produce a tangible result. Applicant respectfully submits that claims 18-34 are not directed to processes. Annex IV(a) of the Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility states, "When a computer program is claimed in a process where the computer is executing the computer program's instructions, USPTO personnel should treat the claim as a process claim ... When a computer program is recited in conjunction with a physical structure, such as a computer memory, USPTO personnel should treat the claim as a product."

With regard to amended claim 18, Applicant respectfully submits that while a computer system is recited that executes the COBOL program, the execution of the COBOL program is not claimed in a process. Rather, the COBOL program is claimed to execute as part of a system including computer components of a COBOL extension layer, the COBOL program, and a computer system. Assuming arguendo, Applicant respectfully submits that claim 18 does produce a useful, tangible, and concrete result. In particular, claim 18 produces the result of the functionality of the COBOL program, namely the performance of a distributed and asynchronous processing task through the execution of the COBOL program. Dependent claims 19-22 are therefore also directed to statutory subject matter.

With regard to amended claim 23, Applicant respectfully submits that the compiler is recited in conjunction with the physical structure of a computer-readable

medium and as such should be treated as a product claim. Assuming arguendo, Applicant respectfully submits that claim 23 does produce a useful, tangible, and concrete result. In particular, claim 23 produces the result of compiled COBOL programs that perform asynchronous and distributed processing tasks upon execution. Dependent claims 24-34 are therefore also directed to statutory subject matter.

With regard to amended claims 1 and 35, Applicant respectfully submits that the methods or processes do produce a useful, tangible, and concrete result. In particular, claims 1 and 35 produce the result of the functionality of the COBOL programs, namely the execution of the COBOL programs results in the performance of distributed and asynchronous processing respectively. Dependent claims 3-17 and 37-39 are therefore also directed to statutory subject matter.

Should further discussion of the statutory merits of the claims be necessary, Applicant respectfully requests the Examiner contact the undersigned.

Applicant Initiated Interview

Applicant thanks Examiner Eric Kiss for his time and consideration of the proposed amendments presented in the telephone interview on May 15, 2007. The proposed amendments to claim 1 are representative of the topics discussed in the interview. In the proposed claim amendment to claim 1, Applicant attempted to clarify that the claimed technical layer enables the distributed processing module through bit-level operating system calls. Examiner Kiss indicated that as broadly as claimed, the PERCobol reference still read on the limitations. Examiner Kiss indicated that while the PERCobol compiler performed many translation and compilation operations on COBOL

programs that eventually object code or machine code would be created that would enable distributed processing through bit-level operating system calls. The proposed claim amendment to claim 1 further attempted to clarify that the COBOL program is being executed. Examiner Kiss raised the question, when does the COBOL program of the PERCobol reference cease to be a COBOL program? A detailed discussion of the amendments presented herein to address these and other points follows.

Response to Rejections under Section 103

In the Office Action dated February 22, 2007, claims 1, 18, 23, and 35 were rejected under 35 U.S.C. § 103(a) as being unpatentable over "PERCobol Evaluation and review Guide: Getting Started" (hereinafter Getting_Started) in view of "PERCobol Enterprise Edition V2.4: Programmer's Guide" (hereinafter Prog_Guide).

The present disclosure is directed to addressing the limitations of the COBOL programming language. Paragraph 006 of the present disclosure states:

> "Unfortunately, COBOL is severely limited in a number of areas compared to the processing techniques available to developers that use other languages such as C or JAVA. POSIX, or Portable Operating System Interface uniX, is a standard UNIX interface for applications to ensure interoperability on equipment from various venders. POSIX includes well know functionality available in programming languages such as C and JAVA for accomplishing distributed and asynchronous processing, such as shared memory, memory and message queues, threads, semaphores and mutexes, events, signal handlers, and sockets."

Further, paragraph 008 of the present disclosure states:

> "The processing techniques described above are examples of useful functionality widely available to programmers using distributed and asynchronous processing languages, such as C and JAVA, but unavailable in COBOL. Frequently, it is desirable for business processes employing COBOL applications to accomplish distributed and asynchronous processing. Although the COBOL language has limitations, it is difficult for businesses with a significant investment in COBOL programs to justify abandoning the COBOL applications and redeveloping the

> applications using a more modern and flexible language, such as C or JAVA. Instead, COBOL systems are typically provided with an interface or "hook" to enable the COBOL program to cooperate with, for example, C or JAVA programs. The C or Java program then performs the distributed and asynchronous processing tasks that the COBOL application is otherwise incapable of handling independently."

The present disclosure utilizes a technical layer that defines bit level mappings of calls to functions of an operating system so as to interface with the operating system in order to enable COBOL programs to perform asynchronous and distributed processing tasks. In an embodiment, the technical layer may be implemented as a library of callable routines that are linked to the COBOL programs. In other embodiments, the technical layer may be implemented as a pre-compiler or a compiler, for example. The technical layer is described in detail in paragraphs 026-039 wherein a discussion of the bit level mapping of calls is discussed in paragraphs 041-046.

With the technical layer defining how to interface with functions of the operating system, COBOL programs may employ the technical layer in the same runtime environment such that additional interfaces with other high level programming languages are not necessary. This enables COBOL programs to be executed in their native COBOL runtime environment so as to take advantage of the inherent efficiencies of COBOL programming while also being able to perform distributed and asynchronous processing tasks.

PERCobol® produced by LegacyJ® is similarly directed to addressing the limitations of the COBOL programming language. The preface on page iii of the Prog_Guide discloses, "Programming with PERCobol adds features that extend standard COBOL-85 and offers functional capabilities which are not defined as part of

the standard". Similarly, page 3 of Getting_Started discloses, "PerCobol compiler and runtime bring together the capabilities of COBOL and Java technology."

Page 3 of Getting_Started discloses, "In its simple form PERCobol ... enables COBOL programs to ... execute in the Java Virtual machine." Unlike the present disclosure PERCobol is directed to interfacing with JAVA as discussed in paragraph 008 of the present disclosure. Specifically, page 8 of Getting_Started disclosed, "The PERCobol compiler compiles, or transforms, COBOL source code into Java executables." Page 10 of Getting_Started describes the transformation process. A COBOL source code program is translated in PERCobol to produce a set of Java source code files. The Java source files are then compiled into an executable Java .class file. As described on page 11 of Getting_Started, the executable Java files may be executed in any Java runtime that includes the percobol.jar Java archive file in the classpath.

Translating COBOL programs for execution in a Java runtime environment may reduce the inherent efficiencies of COBOL programming. Further, the COBOL programs will not be executed in their native environment such as existing batch or CICS environments, but would rather be executed in an environment that supports Java such as running under the UNIX shell on OS/390. This complicates the development of COBOL programs through requiring developers to understand the new environment in which the Java would be executed in.

These and other differences between PERCobol® and the present disclosure are discussed in relation to the claim limitations as follows.

Claim 1:

I. Neither Getting Started nor Prog_Guide disclose a COBOL technical layer enabling a distributed processing module through a call made to a function of an operating system.

Page v of Prog_Guide discloses that functions that are not directly available to COBOL are enabled through Java API functions. As discussed above, calling functions of the operating system enables the COBOL program to perform non-native COBOL functions without requiring an interface with other high level programming languages. This enables the COBOL program to be executed in its native runtime environment. By requiring that non-native COBOL functions be called through Java API calls, PERCobol® requires that developers have knowledge of both COBOL and Java.

II. Neither Getting Started nor Prog_Guide disclose that the COBOL technical layer define a bit level mapping of the call to interface with the operating system.

While page v of Prog_Guide discloses that with native COBOL compilers it is generally possible to call operating system functions to perform functions that are not directly available to COBOL, Prog_Guide does not disclose defining a bit level mapping of the call to the operating system as required by the claims. For example, paragraph 045 of the present disclosure describes exemplary COBOL code that defines the bit level mapping required to call a function of the operating system to establish a socket connection. In the interview, Examiner Kiss indicated that even with the Java code, eventually object code or machine code would be created that would enable distributed processing through bit-level operating system calls. While this generally may be true,

Applicant respectfully submits that this is not disclosure of a technical layer that defines a bit level mapping of the call as currently required by the claims.

III. Neither Getting_Started nor Prog_Guide disclose executing a COBOL program.

As discussed above, Examiner Kiss raised the question, when does the COBOL program of the PERCobol reference cease to be a COBOL program? Applicant respectfully submits that the COBOL programs in PERCobol® cease to be COBOL programs when they are translated into Java source code. Applicant submits that once the program is described in Java source code it becomes a Java program. As such, PERCobol® does not execute COBOL programs, but rather translates COBOL programs into Java programs and executes the Java programs.

IV. Neither Getting_Started nor Prog_Guide disclose the COBOL program and the technical layer operating in the same runtime environment.

Getting_Started discloses on page 10 that any Java runtime may be used if it includes the percobol.jar Java archive file in the classpath. As such, the Java program produced from the COBOL program may operate in a different runtime environment from the PERCobol® compiler. Further, as mentioned above, the COBOL program ceases to be a COBOL program when it is translated into Java source code. As such, while the Java program might operate in the same runtime environment as the PERCobol® compiler, the COBOL program does not operate in the same runtime environment. In particular, the COBOL program is never executed and as such does not operate in any runtime environment.

Dependent claims 3-17 are similarly not disclosed by Getting_Started or Prog_Guide for at least the reasons detailed in I-IV above.

Claim 18:

Claim 18 includes limitations similar to those discussed in Claim 1. The arguments presented in I-IV are herein repeated for claim 18.

Dependent claims 19-22 are similarly not disclosed by Getting_Started or Prog_Guide for at least the reasons detailed in I-IV above.

Claim 23:

Claim 23 includes limitations similar to those discussed in Claim 1. The arguments presented in I-IV are herein repeated for claim 23.

Dependent claims 24-34 are similarly not disclosed by Getting_Started or Prog_Guide for at least the reasons detailed in I-IV above.

Claim 35:

Claim 35 includes limitations similar to those discussed in Claim 1. The arguments presented in I-IV are herein repeated for claim 35.

Dependent claims 37-39 are similarly not disclosed by Getting_Started or Prog_Guide for at least the reasons detailed in I-IV above.

Conclusion

Applicant respectfully submits that the present application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: May 22, 2007

Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANT

CONLEY ROSE, P.C.
5700 Granite Parkway, Suite 330
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)